Payment Gateway API

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Information Technology

By

Arpit Thakur (181411)
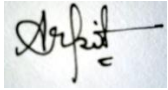
Under the supervision of

Dr. Rajni Mohana

to

Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Table of Contents
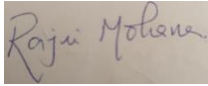
Candidate's Declaration

We hereby declare that the work presented in this report entitled **"Payment Gateway API"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of our own work carried out over a period from January 2022 to May 2022 under the supervision of **Dr. Rajni Mohana** (Associate professor, CSE Department).The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Arpit Thakur, 181411

Certificate By Supervisor

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Rajni Mohana

Associate Professor

Computer Science and Information Technology

Dated: 09/05/2022

Acknowledgement

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we feel a deep sense of gratitude towards them.

We are very grateful to Ms. Rajani Mohana, for providing us an opportunity to do the project work and giving us all support and guidance which enabled us to complete the project duly. We are extremely thankful to her for providing such a great support and guidance, although she had busy schedule managing the university affairs.

We are thankful and consider ourselves fortunate enough to get constant encouragement, support and guidance from our family and friends who also contributed to the successful completion of our project work

Arpit Thakur, 181411

Abstract

The records era (IT) has revolutionized the diverse factors of our lives; specifically it has provided an easy way to go for digital payments. during the Demonetization period, the government of India compelled the human beings at once or indirectly to do all industrial transactions through virtual mode. The common people started to move from conventional payment system closer to virtual bills systems which ensured secure, at ease and convenience. With large technological leaps inside the clever phone and clean net access has led Indian market to simply accept digital bills. the proportion of the virtual payments via other modes is likewise increasing in a sizable speed. The objective of the existing examine is to recognise approximately the numerous types of digital price transactions which are used by the commonplace people in their day-today's lives. This have a look at is mainly primarily based on Secondary statistics. result shows that the virtual revolution has made the cash much less transaction an clean one. As a result that in 2015-2016, a total of 4018 billion has been transacted through cell banking whilst in comparison to 60 billion in 2012-thirteen.The reach of cellular community, net and energy is likewise expanding digital bills to remote areas. So, it's far definitely said that destiny transaction device is cashless transaction.

The widespread adoption of virtualization, microservices, and cloud-native technologies has made API-based interaction between service entities the standard. Due to commercial needs, many platforms still maintain a huge number of obsolete APIs. Simultaneously, many new APIs are increasingly becoming available. If your API communicates with a third-party app, you should be curious in how that app distributes data to the Internet. For example, you might not be concerned about someone learning the contents of your refrigerator, but you might be if that individual can use the same API to locate you. So, the one of the important things to give an attempt to have a secure adaption of the API.

1. Chapter-1: INTRODUCTION

1.1   Introduction

Patron spending via the internet is growing at a sizeable fee. on-line spending these days noticed global double-digit increase nearing 50% on a 12 months-to-year foundation, and inside the India by myself spending is anticipated to reach $three.5 trillion in 2006 [Rob & Opara, 2003, p. 1]. Those traits are fueled by using worldwide economic enlargement. They can be expected to retain in the future. The boom in spending on the net, collectively with the underlying want for comfy transactions, will increase the significance of on-line fee structures. on-line charge systems can be broadly described because the means and techniques worried in carrying out transactions online; however, this description may be improved to encompass the online financial connections among sellers, buyers, economic establishments, and intermediaries. online fee systems had been round for numerous years, yet at the moment are becoming ubiquitous with the increased common use of the net. a number of the advantages furnished by on-line fee systems encompass improved coins go with the flow efficiency, guaranteed transactions, decreased expenses, expanded protection of touchy records, and multiplied safety of the fee provider. given that fraud is a conventional issue with on-line transactions, comfortable on line price structures are mainly important. in spite of the growth and importance of on-line fee within the contemporary international economy, little instructional literature exists on this place that integrates the disparate research streams about on line fee structures and describes their implementation. in addition, online price approaches are largely not noted by means of conventional textbooks, but it's miles important for college students to understand on line payment because it represents the most extensive shift in charge in the last hundred years.

Benefits of Global Payments Integrated's APIs

Worldwide sending and receiving of money included gives strong, bendy APIs which might be clean and fast to combine, no matter one's dynamic approach. It decreases

velocity to mart and eases the improvement circulations of builders. I desperately wants the potential to have a better architecture for the application program interface to have. For the instance, we have our cellular structures, we are in the position to ship, acquire, and conceive with the application program interface without any error. Below is the working:

utility program Interface Request – on this element, the software itself sends all the necessity records and statistics to the protected software program improvement package.

Processing of the Request – The way a charge software program interface generally works is that an utility transmits a message request to the API, which relays it to the fee processing community, that provides a local consumer interface for customers to pick out or upload a payment technique, a transport cope with, a shipping option, and phone data in an clean, fast, and cozy way, which in turn procedures the request and responds to the API, which then relays the response to the software.

capture results - The charge software Programming Interface, additionally called the payment Gateway API or fee Processing API, works to attach the charge option to another, existing utility, consisting of a enterprise checkout feature, to the fee system. users who make purchases on the developer's app will pay without problems and comfortably.

replica receipt and sent e-affirmation -The satisfactory answers we can get is the output of the charge that we have come upon in these types of strategies as an e-confirmation.
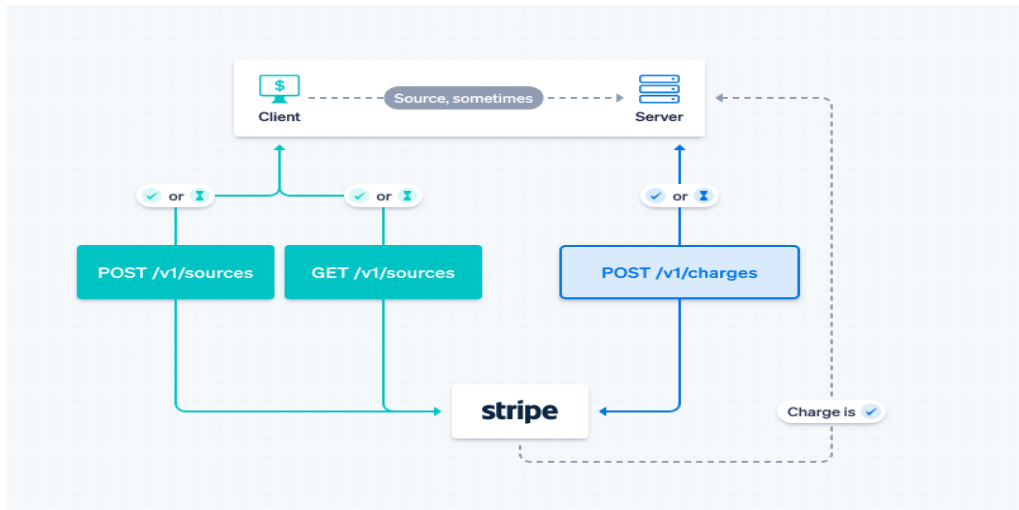
Figure 1. Payment API model

1.2   Problem Statement

The Task of this project is to come up with the designing and developing the online Payment API which will be east to test and maintain. And the API should be able to provide the services to the various online purchasing e-commerce companies and more.

An utility programming interface is classical records that allows two devices, databases or programs to connect and proportion statistics. In other words, it's the language backstage software program that makes it clean for us to do things online or with our smartphones.

We use the API all of the time, whether or not we know it or no longer. as an instance, when we buy airline tickets thru a 3rd-birthday celebration tour reservation web site, the API authorizes verbal exchange between the airline and the tour company. Or if we've got a software development kit thermostat in our domestic, the API allows communication that allows it to modify the temperature remotely in an character room.

The Task of this project is to come up with the designing and developing the online Payment API which will be east to test and maintain. And the API should be able to

3

provide the services to the various online purchasing e-commerce companies and more.

1.3 Objectives

- The goal of the undertaking is to offer a fee gateway API version to correctly test and maintain the API and make the API capable of work smoothly. in this work, we gift a semi-supervised gaining knowledge of method that outperforms, even greater, the performance of API in terms of the modelling of the API and the relationship between the two APIs which is established thru the relaxation Template. There are distinctive functions that can or won't be to be had, depending at the API and processing platform. as an example, there's the capability to save transactions offline and feature them processed whilst capability is restored. Or the API can be able to aid a diverse array of diverse transaction types past the fundamental card manufacturers and not unusual charge strategies.

- furthermore, a rate API can be able to consolidate all the statistics transmitted via the platform, thereby providing useful statistics that has the potential to inform business organization and advertising and advertising and marketing choices.

- The security is one of the most important aspect of any API or a website. For authentication purposes, it is also necessary to use some technology to authenticate the user in the APIs. For this objective, I have used the hashing algorithms in both the APIs.

1.4 Methodology

The idea is to build an API for the Payment API Gateway. For this purpose, there is two separate APIs built to perform the complete two different services. The different service would be :

User_Service : It's main tasks would be

- create a new user.
- Get all the users
- Get user by Id.

Wallet_Service : It's main tasks would be

- Update the wallet.
- Create Wallet.
- Get wallet, if required.

Transaction_Service : It's main tasks would be

- We should be able to send money from one user to another.
- We should be able to get the money.
- We should be able to sent transaction to the user if he wants on their mail.

One fundamental gain of a rate API is that it may beautify a client's shopping enjoy through doing away with the want to fill out checkout bureaucracy, a tedious challenge - especially on a telephone - that may result in shopping cart abandonment. thanks to the capacity to reuse stored rate information, customers can complete a purchase on a cell device with only a few taps.
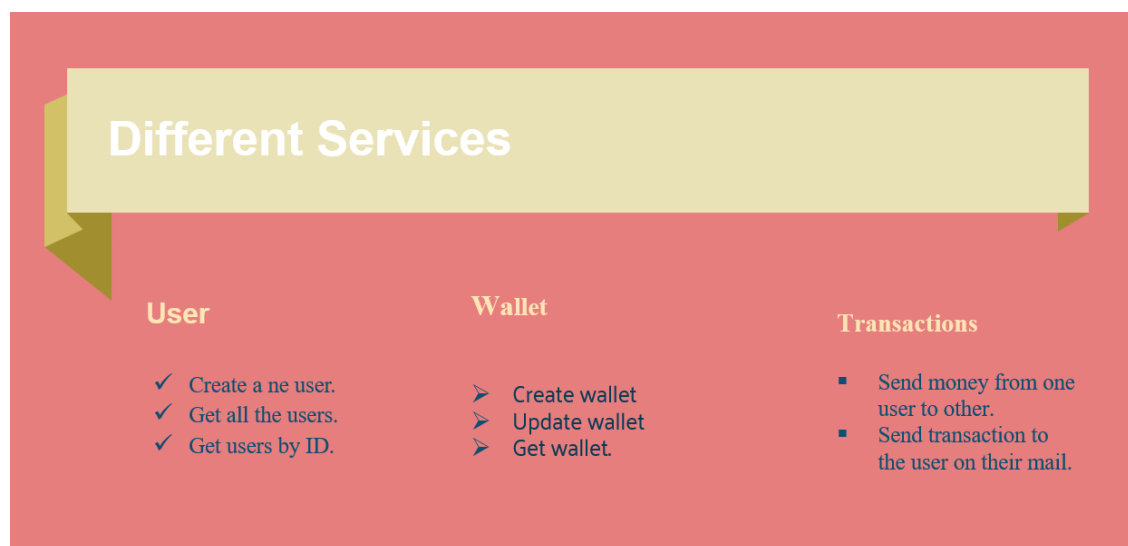

Figure 2. Different services to be used in the API

1.5   Organization

This Project is completely based on backend working. The main methods, technologies and tools that are associated with this project are:

Methods Used:

- Microservice Architecture
- Object Relational Mapping
- Auto Configuration
- Hashing Algorithm
- Authentication

Technologies Used:
- Java
- SpringBoot
- Swaggers
- Hibernate
- Rest Template
- Actuators
- MySQL

Every of the technologies are freely to be had and the technical capabilities required are attainable. Time obstacles of the product development and the benefit of implementing the usage of those technologies are synchronized. From these it's clean that the assignment is technically feasible.

2. Chapter-2: LITERATURE SURVEY

Price Gateway APIs

A fee gateway API integrated with present virtual methods can connect a checkout system to a fee community.

Advantages of a price Gateway API

Figure 3. Benefits of payment Gateway API

The widespread adoption of virtualization, microservices, and cloud-native technologies has made API-based interaction between service entities the standard. Due to commercial needs, many platforms still maintain a huge number of obsolete APIs. Simultaneously, many new APIs are increasingly becoming available. If your API communicates with a third-party app, you should be curious in how that app distributes data to the Internet. For example, you might not be concerned about someone learning the contents of your refrigerator, but you might be if that individual can use the same API to locate you. So, the one of the important things to give an attempt to have a secure adaption of the API.

One of the most critical aspects of any API or website is security. It is also required to employ some technique to authenticate the user in the APIs for authentication reasons. I used the hashing techniques in both APIs to achieve this goal.
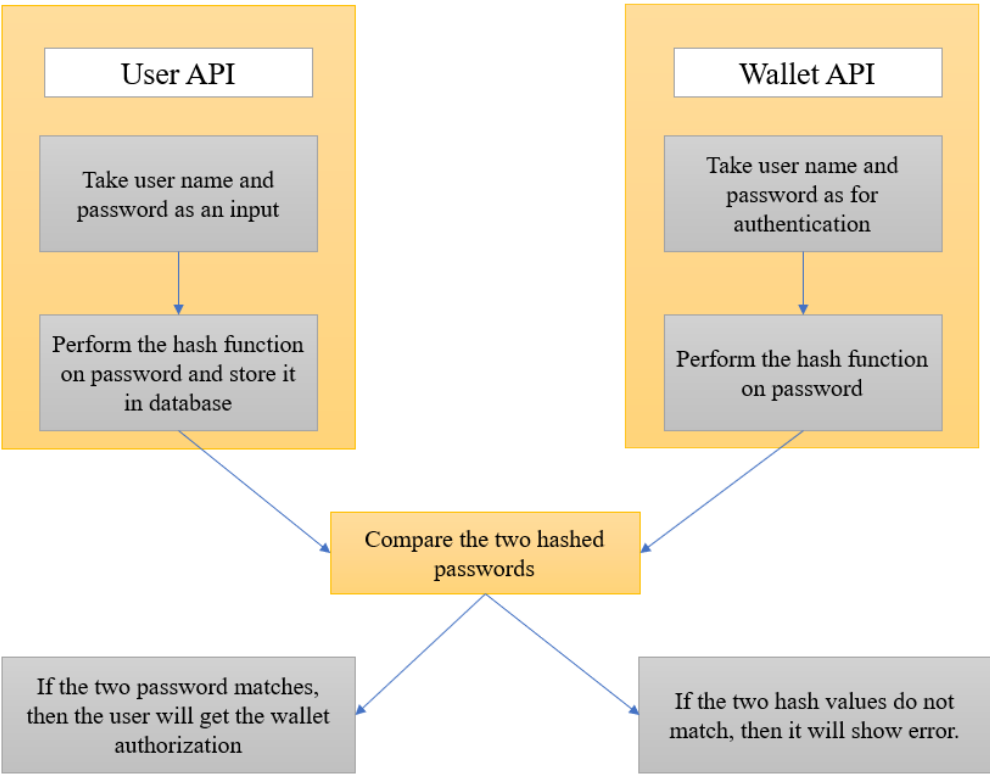

Figure 4. flow chart of the authentication implementation

The number one advantage of using a unfastened gateway API is that it could simplify the price technique (request, authorization, receipt and affirmation). The way a charge API typically works is that an utility transmits a message/request to the API, which relays it to the price processing network, which in turn procedures the request and responds to the API, which then relays the response to the application.

stepped forward person experience:

One essential gain of a charge API is that it is able to enhance a customer's purchasing revel in by way of manner of casting off the want to fill out checkout paperwork, a tedious task - mainly on a cellular phone - that may reason buying cart abandonment. thanks to the functionality to reuse stored charge statistics, customers can whole a buy on a mobile device with only a few faucets.

Innovation:

each other advantage of a fee API is that it allows a service provider to innovate. typically, a essential API rate integration is simple, smooth to check and without difficulty performed. Or the API can be capable of guide a diverse array of various transaction sorts past the important card manufacturers and commonplace charge techniques.

furthermore, a fee API can be capable of consolidate all the data transmitted through the platform, thereby supplying beneficial records that has the ability to inform business and marketing   choices

but if your business enterprise has current thoughts - and a developer who has the technical realize-the way to do the customization - an API can be custom designed to your heart's content material cloth, thru integrating out of doors applications, as an instance. inside the intervening time, a payment API likely permits a business agency to maintain constant branding and consumer experience at some stage in channels

security and Fraud Prevention:

ultimate, however certainly now not least, some other core gain of a charge API pertains to protection and fraud prevention, which in flip eases regulatory compliance. That's due to the fact consumer information and fee information is commonly stored through the host (now not the provider provider), keeping that touchy information out of the hands of a service provider. This makes it less tough for a carrier issuer to live PCI compliant and EMV®-ready at a time at the same time as failing to perform which can result in great exposure to monetary liability.

additional advantages:

There are different features that could or might not be to be had, relying at the API and processing platform. for instance, there's the capability to shop transactions offline and feature them processed when functionality is restored. Or the API may be able to assist a various array of various transaction sorts beyond the fundamental card manufacturers and commonplace payment methods.

moreover, a price API can be able to consolidate all of the facts transmitted through the platform, thereby providing helpful records that has the ability to inform enterprise and marketing selections.

similarly, software vendors have determined that integrating a price API into its software program has the capability to offer an incremental sales circulation via revenue sharing with the price processor.

blessings of world payments incorporated's APIs

global bills incorporated gives strong, flexible APIs that are smooth and fast to integrate, no matter your integration approach. This reduces velocity-to-marketplace and eases the development cycles for builders. We also have the potential to do the development for you if that is the path you prefer to take.

for example, for our mobile answers, you are able to ship, receive, and connect with our API seamlessly. here's the way it works:

- ship API Request - Your utility sends all non-touchy parameters to the embedded SDK.

- system the Request - the worldwide bills included API communicates with the PIN pad tool to activate for card insertion, faucet, swipe or guide access, before accumulating encrypted card information and performing an authorization request through the worldwide payments incorporated Platform.

- capture consequences - the global bills incorporated SDK will provide you with the non-sensitive price consequences together with the bank approval code, final 4 card digits, and an encrypted token that can be used for subsequent payments.

- Print Receipt or send eConfirmation - Our answer offers your application the flexibility of showing or sending payment confirmation to the client.

Transactions person Interface (Transactions UI) From global bills included

ISVs can now take the complexities out of their charge integration with our new Transactions user Interface (Transactions UI). This interface allows ISVs to make a unmarried API call to guide more than one charge sorts, as well as break up the fee amounts into numerous transactions for a easy, unmarried omni-channel enjoy.
This lets in ISVs to stay inside their environment, programming language, and operating system whilst speeding up time to marketplace, permitting international expansion and increasing product differentiation.

Transactions UI functions

- All functionality via one API without the complexities of custom integrations

- New value-added products and functions in the destiny with little to no improvement effort
- Cloud-primarily based carrier to monetize bills and scale to a broader patron base

benefits to ISVs' clients

- decreased time and effort required for accepting bills
- Simplified reconciliation with the addition of a new coins smooth kind and reporting
- Streamlined bills workflow with an smooth to use interface for taking pictures transactions
- accelerated likelihood of customers buying goods and services with the capability to accept multiple price strategies inside the equal transaction

To analyze greater about the blessings of global bills included's APIs or our new Transactions user Interface (Transactions UI), contact us these days.

3. Chapter-3: SYSTEM DEVELOPMENT

Digital commerce reports are increasingly more stepping out of doors the boundaries of the legacy computer-first web sites. To make more recent touchpoints consisting of cell apps, in-keep digital, interactive show, wealthy content material, and IoT natively shoppable or assist one-touch, gesture or voice-enabled transactions, traders want to increase price processing everywhere.

The key to reaching omnichannel trade is API-pushed systems that allow applications to share and manner information. Many companies have followed ecommerce microservices as a bendy way to increase, customize and scale their trade functionality to suit the new world of shoppable touchpoints.

Microservices are small programs constructed around precise business capability decoupled from different pieces of your commerce platform. Their independence and committed **APIs** permit them to integrate with numerous touchpoints and be scaled, up to date, and removed with out impacting website overall performance, the database, or code of your middle commerce platform. They're specifically useful when you want to fulfill specific use cases without the chance of "breaking your build."

3.1 *Design:*

To build a microservice architecture or to design a microservice architecture we need to make a different API for every different service in the application we are going to include.

The two services have to run as two different APIs but have to be connected through a gateway, which can help an API to borrow some data from another API . Both have to be connected through a single gateway. For this purpose , the Rest Template comes into play. The use of Rest Template is done to connect the functionality of two APIs being separated.
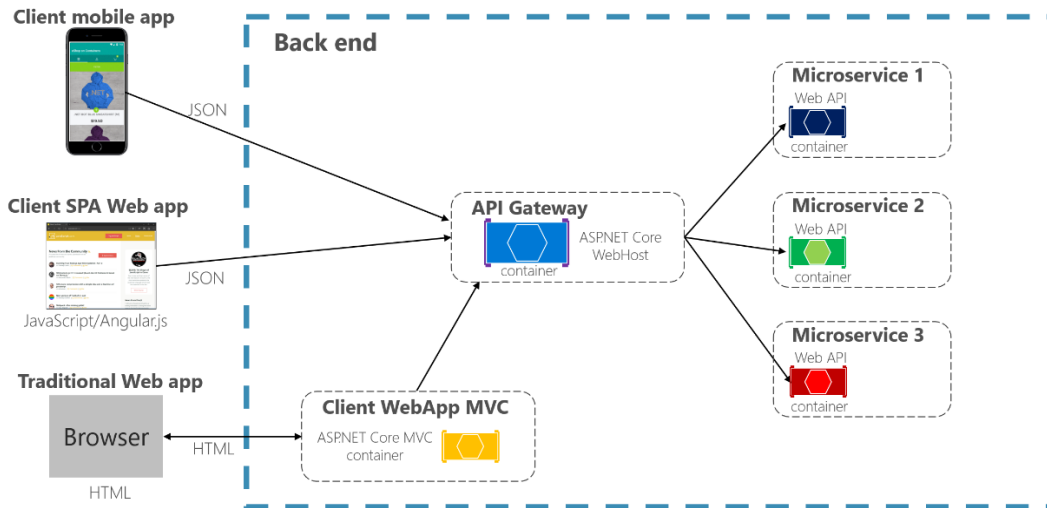
Using a single custom **API Gateway service**

Figure 4. Design of an API in microservice architecture

In this case, beginning from the pinnacle left nook, a user sends an occasion or message to execute a payment as an access point. The customers can be cellular, internet, or any outside tool / utility that acts as the entry factor with the groups payments structure.

This request to execute bills connects to your services thru the bills API. this is the bridge to the internal relevant payments occasion streams, where streams are managed to decide what selection or sub-choice of moves need to be taken. For practical functions, we'll proceed thru this architecture as if all alternatives are necessary to make sure coverage of all elements and offerings.

the first motion taken is validation of the incoming bills request, the usage of the validation microservices presenting integration to all wanted systems in an corporation to validate finances, customers (users), and more. as soon as validation is completed a message is sent back to the payments event streams for similarly processing.

the following two levels of the bills architecture are not constantly necessary as they rely upon validation consequences. If there are any issues or flags raised on a payment request then the subsequent step could be to trigger using the anti-cash laundering (AML) microservices or fraud detection microservices. every of those collections decide if action desires to be taken, if payment requests want to be blocked, if the consumer wishes to be

suggested, and in all likelihood blocking funds. both of these service regions are leveraging information caching to ensure current information is available to the selection control tooling used to lower back those processing levels. Any consequences are then pushed lower back to the bills occasion streams for in addition processing.

Following the procedure elements to the proper we're going to arrive on the clearing microservices in which processing for actual payment budget planning wherein accounts are debited earlier than routing the price range to the requested events. these outcomes are sent back to the bills event streams for in addition processing.

subsequently, routing microservices are accessed to ensure the price range from the processed bills are distributed to the indicated parties thru the to be had bills community. notice that the bills community is shown as an external secure cloud element, supposed to suggest only that it's an external network and dependent on the place the answer is being deployed in for specifics as to the connection and information protocols used.

the second discern at the proper here is labeled as a direct bills facts instance and is meant to offer greater insights into the motion of event streams and glide of statistics through the above described system of executing on a bills request. Exploring the information of this figure is left to the reader.

3.2   *Designing Procedure:*

- Start
- Go to the Spring Initializr website.
- Include the dependencies like MySQL Driver, JPA /Hibernate, Spring web, etc.
- After that click on build Spring Boot application.
- Then, open the project in IntellIJ.
- Then build controllers, model of the API, Repositories, and Services.
- Then connect the MySQL database to the application using JPA/Hibernate.
- Then write GetMapping , PutMapping, etc. to perform functions in the API.
- Built another API with same Configuration.
- Then, use Rest Template to connect between two or more APIs.
- End

3.3 *Model Development:*

Microservices architecture scalability is terrific for permitting aid for a huge variety of platforms and devices as internet, mobile and IoT backend can be deployed, constant, and up to date independently of every different. Microservices are small and flexible, which means that you don't should have interaction with a large code base on every occasion your crew wants to make a change within the code.

Alternatively, development groups can choose exactly the portions of code they need to engage with and enhance each productiveness and maintainability. eventually, it promotes the idea of groups reusing each other's code as opposed to developing independently and from scratch.

Microservices structure is especially vital for DevOps groups because it gives accelerated agility so the group can roll updates out lots faster--and extra tailor-made to every crew's timeline--because of shorter build, take a look at, and installation cycles. Reliability is also increased seeing that an issue with one microservice only influences that microservice, no longer the complete application. This substantially reduces any opportunity of the complete software failing.

Microservices are also extraordinarily modifiable, providing teams the ability to apply new frameworks, libraries, and records resources. the main concept is that microservices architecture intentionally leverages agile concepts, in which the development attempt is unfold across go-purposeful teams that work extra independently.

Classification Setup: -

The idea is to build an API for the Payment API Gateway. For this purpose, there is two separate APIs built to perform the complete two different services. The different service would be :

User_Service : It's main tasks would be
-create a new user.
-Get all the users
-Get user by Id.

Wallet_Service : It's main tasks would be
-Update the wallet.
-Create Wallet.
-Get wallet, if required.

Transaction_Service : It's main tasks would be
-We should be able to send money from one user to another.
-We should be able to get the money.
-We should be able to sent transaction to the user if he wants on their mail.

One major gain of a charge API is that it can enhance a consumer's purchasing experience via casting off the want to fill out checkout forms, a tedious venture - mainly on a smartphone - which can lead to purchasing cart abandonment. thanks to the ability to reuse saved charge statistics, clients can entire a purchase on a cellular tool with just a few faucets.

Figure 5. Different services to be used in the API

The below are the some glimpses of the code written to make an API functional :

Figure 6. Code for Controller class

The above class is a controller class. The most of the functions or the requests which are passed to the API are functional here. The controller class is for the user_service API.

1. The first request in above code if the @GetMapping , which returns all the users to the client in the form of JSON or XML.
2. The second request is the @PostMapping , which is to register a new user into the MySQL database.
3. The third and the last request is the @GetMapping for finding the user by id.

Whenever the function is called, the controller is called by the main class to function, and the identity of the controller class is recognized by the @RestController annotation of the java class.


```java
package com.example.user_service.Model;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;

@Entity
@Getter
@Setter
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    @Column(name="name")
    private String name;

    @Column(name="email")
    private String email;

    @Column(name="mobile")
    private String mobile;

    @Column(name="is_kyc_done")
    private boolean is_kyc_done;
}
```

Figure 7. Model class of the API

The above class is the model class named User.java. The structure of an API and data consumed by the API is all designed in this class. @Entity is used for Object Relational Mapping in this class.

The below image is the complete structure of the User_Service API .The controller and model class is discussed above. The Repository class is the class which connects the MySQL database to the application .The functionality of each package is given below:

1. Controller- It contains classes for request mapping.
2. Exception- It contains classes for exception handling in the project.
3. Model- It contains the classes defining the model structure of the API.
4. Repository- It contains the classes which connects the database with the particular model class.
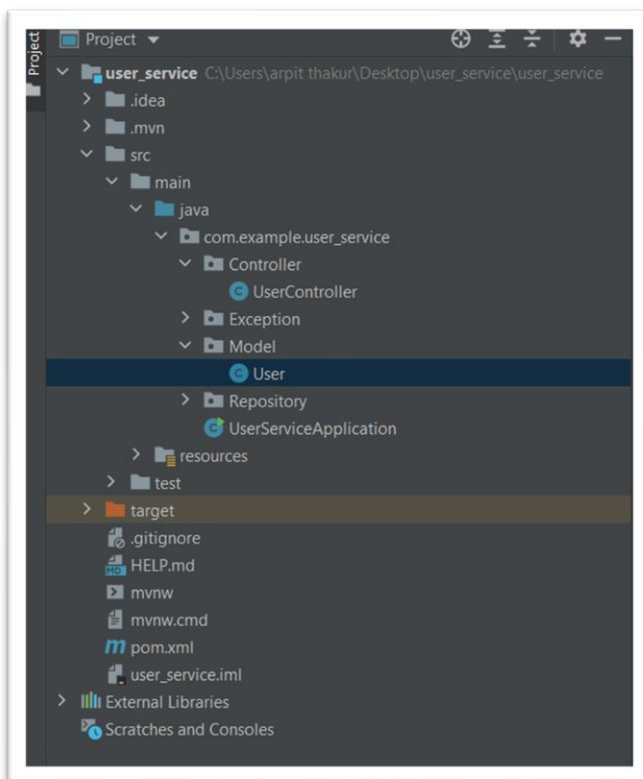
Figure 8. Project Structure of an API

One of the most critical aspects of any API or website is security. It is also required to employ some technique to authenticate the user in the APIs for authentication reasons. I used the hashing techniques in both APIs to achieve this goal.
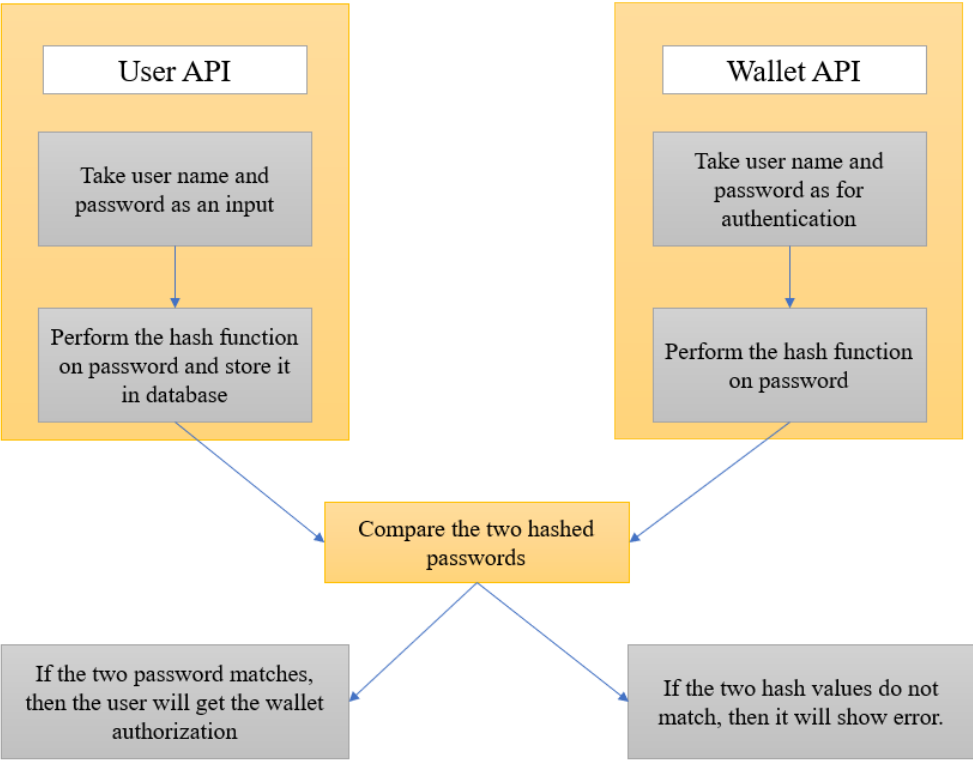
Figure 9. flow chart of the authentication implementation

Above shown flow chart is the description of the model that I have used in the microservice architecture APIs that I have designed.

4. Chapter-4: PERFORMANCE ANALYSIS

4.1 Analysis

The analysis of the project is that , the two APIs are very easy to update because it reduces the complexity of the project, as we have built the two APIs separately in a microservice architecture. The payment API works smoothly and fine.

4.2 Results

We have used the two of the requests as a sample in the API and both are working fine.
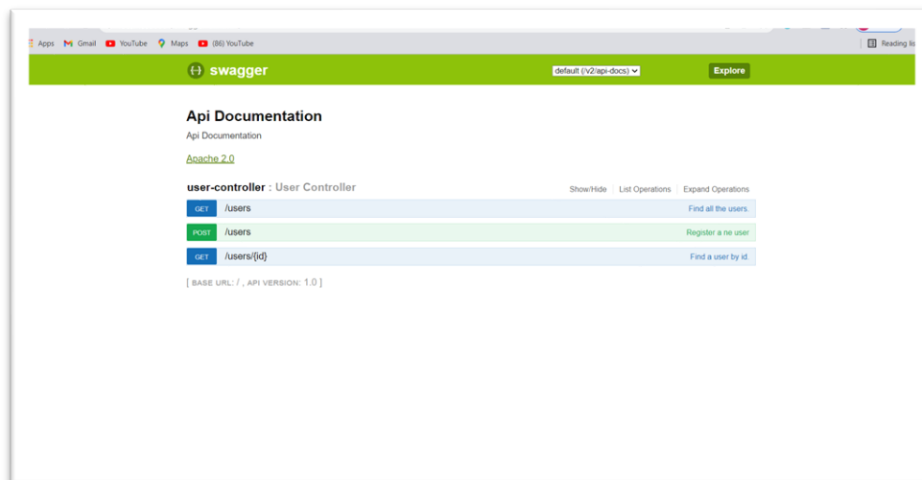
Figure 10. Working Swagger Application on Chrome

4.3 Screen shots of the various stages of the Project

The below are the some glimpses of the code written to make an API functional :
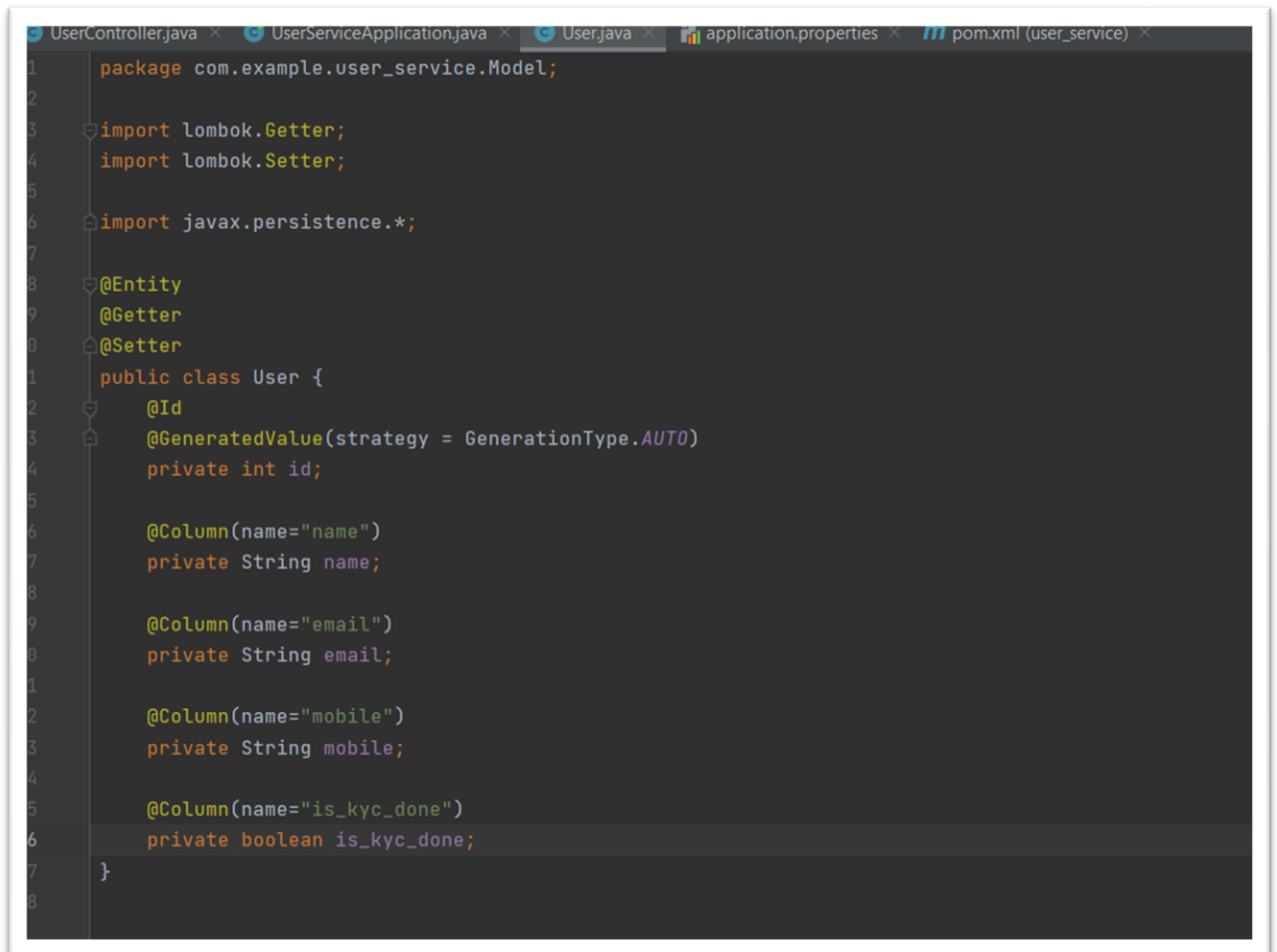
```java
     import java.util.List;
     import java.util.logging.Logger;

     @RestController
     public class UserController {
         private final static Logger LOGGER = Logger.getLogger(Logger.GLOBAL_LOGGER_NAME);

         @Autowired
         private UserRepository repository;

         @ApiOperation(value="Find all the users.")
         @GetMapping("/users")
         List<User> findAll() { return repository.findAll(); }

         @ApiOperation(value="Register a ne user")
         @PostMapping("/users")
         @ResponseStatus(HttpStatus.CREATED)
         User newUser(@RequestBody User newUser) { return repository.save(newUser); }

         @ApiOperation(value="Find a user by id.")
         @GetMapping("/users/{id}")
         User findOne(@PathVariable int id){
             LOGGER.info( msg: "/users/id is called ith id "+id);
             return repository.findById(id)
                     .orElseThrow(()-> new UserNotFoundException(id));
         }
     }
```

Figure 11. Code for Controller class

The above class is a controller class. The most of the functions or the requests which are passed to the API are functional here. The controller class is for the user_service API.

- The first request in above code if the @GetMapping , which returns all the users to the client in the form of JSON or XML.
- The second request is the @PostMapping , which is to register a new user into the MySQL database.
- The third and the last request is the @GetMapping for finding the user by id.

23

Whenever the function is called, the controller is called by the main class to function, and the identity of the controller class is recognized by the @RestController annotation of the java class.


```java
package com.example.user_service.Model;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;

@Entity
@Getter
@Setter
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    @Column(name="name")
    private String name;

    @Column(name="email")
    private String email;

    @Column(name="mobile")
    private String mobile;

    @Column(name="is_kyc_done")
    private boolean is_kyc_done;
}
```

Figure 12. Model class of the API

The above class is the model class named User.java. The structure of an API and data consumed by the API is all designed in this class. @Entity is used for Object Relational Mapping in this class.

The below image is the complete structure of the User_Service API .The controller and model class is discussed above. The Repository class is the class which connects the MySQL database to the application .The functionality of each package is given below:

5. Controller- It contains classes for request mapping.
6. Exception- It contains classes for exception handling in the project.
7. Model- It contains the classes defining the model structure of the API.
8. Repository- It contains the classes which connects the database with the particular model class.
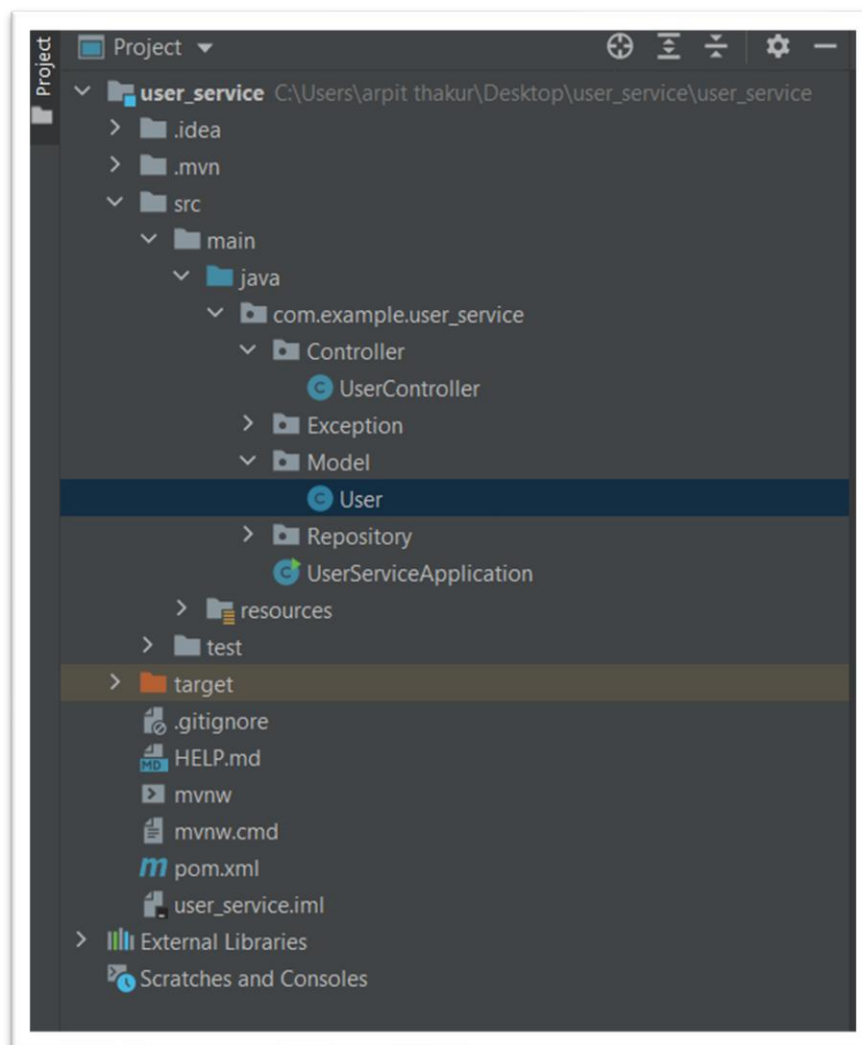
Figure 13. Project Structure of an API

Figure 14. Exception Class

Figure 15. Repository Class using JPA/Hibernate

4.4 Results

The two results of the different mapping are:

@PostMapping: To create a new user.

Figure 16. Input for @PostMapping

Figure 17. Output for the @PostMapping

@GetMapping: To get all the users


```
Curl

curl -X GET --header 'Accept: application/json' 'http://localhost:8081/users'

Request URL

http://localhost:8081/users

Request Headers

{
  "Accept": "*/*"
}

Response Body

[
  {
    "id": 1,
    "name": "Arpit Thakur",
    "email": "thakurap111@gmail.com",
    "mobile": "7888751545",
    "_kyc_done": true
  }
]

Response Code

200

Response Headers

{
  "connection": "keep-alive",
  "content-type": "application/json",
  "date": "Sat, 04 Dec 2021 12:06:37 GMT",
```

Figure 18. Output of the @GetMapping

4.5   Comparisons

As compared to the payment API Gateway project from the research papers, this way of implementing the API in microservice architecture is more efficient. This is because of the two reasons, the firstly one is the code will not be as complex as including all the services in the single API , and the second reason is that, it is easily maintainable and easy to test.

For User authentication , I have used two hashing algorithms one by one and then compared their performance as shown below in the table:

Hashing Algorithm	Case 1 (ms)	Case 2 (ms)	Case 3 (ms)	Case 4 (ms)
SHA-256	737	851	1260	1168
SHA-512	1056	1158	1227	1118

Case 1 : String length of 36 characters
Case 2: String length of 49 characters
Case 3: String length of 85 characters
Case 4: String length of 72 characters

Conclusion of the above shown results is that, SHA-256 is faster in the cases where the length of the string is short and SHA-512 is faster in the cases where the length of the string is long.

5. Chapter-5: CONCLUSIONS

5.1   Conclusions

Worldwide payments included gives strong, bendy APIs which might be clean and fast to combine, no matter your integration approach. This reduces velocity-to-market and eases the improvement cycles for builders. We additionally have the potential to do the development for you if that is the route you favour to take.

As compared to the payment API Gateway project from the research papers, this way of implementing the API in microservice architecture is more efficient. This is because of the two reasons, the firstly one is the code will not be as complex as including all the services in the single API , and the second reason is that, it is easily maintainable and easy to test. Application Programming Interfaces (APIs) permit the seamless integration of software program, making it less complicated than ever for programs to share statistics in actual-time. on the subject of bills, APIs have converted the experience, taking into consideration quicker, more comfortable and lower value transaction processing.

5.2   Future work

RabbitMQ as the dealer in a Microservices architecture
RabbitMQ allows asynchronous processing, which means that it lets in you to place a message in a queue with out processing it straight away. RabbitMQ is consequently best for lengthy-going for walks tasks or blocking obligations, permitting internet servers to reply quick to requests instead of being forced to carry out computationally in depth duties instant. RabbitMQ without a doubt stores messages and passes them to customers when equipped.

Security and Authentication
We can give a parameter as 'password' which would be a password for every user, and further can use a BCryptPasswordEncoder to encrypt that password. This enables

authentication and security within the API. We can also use JSON web tokens for authorization.

5.3   Applications

- Online payment
- E-commerce Applications
- Payment in Play Store
- Banks use API to be used by various applications for payment purpose.

References:

1. -Hong, X. J., Sik Yang, H., & Kim, Y. H. (2018). Performance Analysis of RESTful API and RabbitMQ for Microservice Web Application. *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 257–259. https://doi.org/10.1109/ICTC.2018.8539409

2. Unsal, E., Oztekin, B., Cavus, M., & Ozdemir, S. (2020). Building a Fintech Ecosystem: Design and Development of a Fintech API Gateway. *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–5. https://doi.org/10.1109/ISNCC49221.2020.9297273

3. Zhao, J. T., Jing, S. Y., & Jiang, L. Z. (2018). Management of API Gateway Based on Micro-service Architecture. *Journal of Physics: Conference Series*, *1087*, 032032. https://doi.org/10.1088/1742-6596/1087/3/032032

4. Francia, G. A., & Francia, R. R. (2007). An Empirical Study on the Performance of Java/.Net Cryptographic APIs. *Information Systems Security*, *16*(6), 344–354. https://doi.org/10.1080/10658980701784602

5. Nyarko-Boateng, O., Weyori, B. A., & Tetteh, L. A. (2020). Optimized Authentication Model for Online Transaction Payments. *Journal of Computer Science*, *16*(2), 225–234. https://doi.org/10.3844/jcssp.2020.225.234

6. Jaramillo, D., Nguyen, D. v, & Smart, R. (2016). Leveraging microservices architecture by using Docker technology. *SoutheastCon 2016*, 1–5. https://doi.org/10.1109/SECON.2016.7506647

7. Gómez, O. S., Rosero, R. H., & Cortés-Verdín, K. (2020). CRUDyLeaf: A DSL for Generating Spring Boot REST APIs from Entity CRUD Operations. *Cybernetics and Information Technologies*, *20*(3), 3–14. https://doi.org/10.2478/cait-2020-0024

8. Guntupally, K., Devarakonda, R., & Kehoe, K. (2018). Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example. *2018 IEEE International Conference on Big Data (Big Data)*, 5328–5329. https://doi.org/10.1109/BigData.2018.8621924

9. Velmurugadass, P., Dhanasekaran, S., Shasi Anand, S., & Vasudevan, V. (2021). Enhancing Blockchain security in cloud computing with IoT environment using ECIES

and cryptography hash algorithm. *Materials Today: Proceedings*, *37*, 2653–2659. https://doi.org/10.1016/j.matpr.2020.08.519

10. El-Haii, M., Chamoun, M., Fadlallah, A., & Serhrouchni, A. (2018). Analysis of Cryptographic Algorithms on IoT Hardware platforms. *2018 2nd Cyber Security in Networking Conference (CSNet)*, 1–5. https:https://doi.org/10.1109/CSNET.2018.8602942

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
PLAGIARISM VERIFICATION REPORT

Date: May 11, 2022

Type of Document (Tick): | PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper |

Name: Arpit Thakur Department: CSE Enrolment No 181411

Contact No. 7888751545 E-mail. thakuraplll@gmail.com

Name of the Supervisor: Dr. Rajni Mohana

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): Payment Gateway API

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.
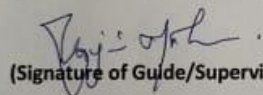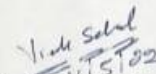
Complete Thesis/Report Pages Detail:
- Total No. of Pages = 33
- Total No. of Preliminary pages = 6
- Total No. of pages accommodate bibliography/references = 2

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....2.0..%....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages		Word Counts	
Report Generated on	• Bibliography/Images/Quotes • 14 Words String		Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com